Western Norway University of Applied Sciences

Project report in DAT255 – Deep Learning Engineering

Comparative analysis of deep learning architectures for sentiment analysis

25.04.2025

h671444

h669796

Section 1: Introduction

1.1 Motivation & Background

In the contemporary digital marketplace, customer reviews exert a significant influence on consumer purchasing decisions and overall brand perception (Chen, Samaranayake, Cen, Qi, & Lan, 2022). Platforms across e-commerce, entertainment, and social media accumulate vast quantities of user-generated text daily. Manually processing this feedback is infeasible at scale. Therefore, automated methods for analyzing sentiment, discerning whether a review expresses a positive or negative opinion, are crucial. Such tools enable businesses to rapidly process millions of comments, gain insights into customer satisfaction, identify areas for product improvement, and implement responsive engagement strategies in near real-time. The ability to accurately classify sentiment automatically unlocks substantial value from unstructured text data.

1.2 Project Goal

The primary objective of this project is to develop, train, and compare deep learning models for binary sentiment classification of product reviews. Specifically, we implement and evaluate a Convolutional Neural Network (CNN), followed by two Recurrent Neural Network (RNN) architectures: one employing bidirectional Gated Recurrent Units (Bi-GRU) and another utilizing a bidirectional Long Short-Term Memory (Bi-LSTM) cells, both in a bidirectional configuration. The models are trained on a substantial subset of the Amazon Reviews Polarity dataset. Beyond achieving high classification accuracy, a key goal is to analyze how specific architectural choices and hyperparameter influence their performance, providing insights into their respective strengths and weaknesses for sentiment analysis.

1.3 Rationale for Deep Learning

Sentiment classification presents challenges that traditional machine learning approaches often struggle to address effectively. Methods like Support Vector Machines (SVMs) or Naive Bayes typically depend on manual feature engineering (like bag-of-words, TF-IDF, etc.) and may fail to capture the nuances of human language, such as word order, context, negation, and sarcasm, like understanding "not bad" as positive, (Pang & Lee, 2008).

Deep learning models excel in this domain by automatically learning hierarchical feature representations directly from raw text data (Young, Hazarika, Poria, & Cambria, 2018). As a starting point, a CNN was implemented, a type of deep learning architecture that has shown strong performance in text classification tasks due to its ability to learn local dependencies and hierarchical features in sequences (Yoon, 2014).

CNNs functions as learnable n-gram detectors and efficiently capture key features irrespective of their absolute position in text. Subsequently, we explore more sophisticated sequential models, specifically Bidirectional GRU and LSTM, which naturally process sequential data, capturing temporal dependencies and contextual relationships between words. Bidirectional RNNs further enhance these capabilities by considering both preceding and subsequent context when interpreting each word.

1.4 Existing Solutions and Project Contribution

The task of sentiment analysis on datasets like the Amazon Reviews Polarity corpus has been extensively studied. Standard benchmark tools, such as fastText, typically achieve accuracies around 90% on the full dataset (Joulin, Grave, Bojanowski, & Mikolov, 2016). More complex, large-scale pre-trained Transformer models like BERT have pushed the state-of-the-art performance beyond 95% accuracy (Xie, Dai, Hovy, Luong, & Le, 2020). While these Transformer models offer high accuracy, they often come with significant computational costs in terms of model size, training time, and inference latency.

The contribution of this project lies in exploring the effectiveness of significantly lighter-weight architectures. Our aim is to achieve competitive performance, targeting accuracy in the vicinity of 90%, but with models that are potentially orders of magnitude smaller and faster than Transformer-based solutions. This makes the models highly practical for deployment scenarios requiring computational efficiency.

1.5 Deployment

To demonstrate practical applicability, a simple deployment was implemented using a lightweight web framework called Streamlit. This interface allows users to input custom review text and receive real-time sentiment predictions (positive or negative). The deployment highlights the importance of inference efficiency, particularly on standard CPU hardware, and serves as a practical comparison point between CNN and RNN architectures, not only in terms of predictive accuracy but also in responsiveness and usability.

Section 2: Data

2.1 Dataset overview

This project utilizes the Amazon Reviews Polarity dataset sourced from Kaggle (Jain, 2025). This dataset is a large-scale corpus specifically designed for binary sentiment analysis tasks. The dataset comprises 3.6 million English-language customer reviews sourced from Amazon.com. A key characteristic of this dataset is its balanced nature, containing an equal number of positive and negative reviews (1.8 million each). Reviews are labeled as '2' for positive and '1' for negative. The reviews span a diverse range of product categories, contributing to the dataset's general applicability. For computational efficiency and practical experimentation, we selected a subset of 200,000 reviews, maintaining the balanced 50/50 split between positive and negative classes. Each review entry includes a polarity label, a review title, and the main review text body.

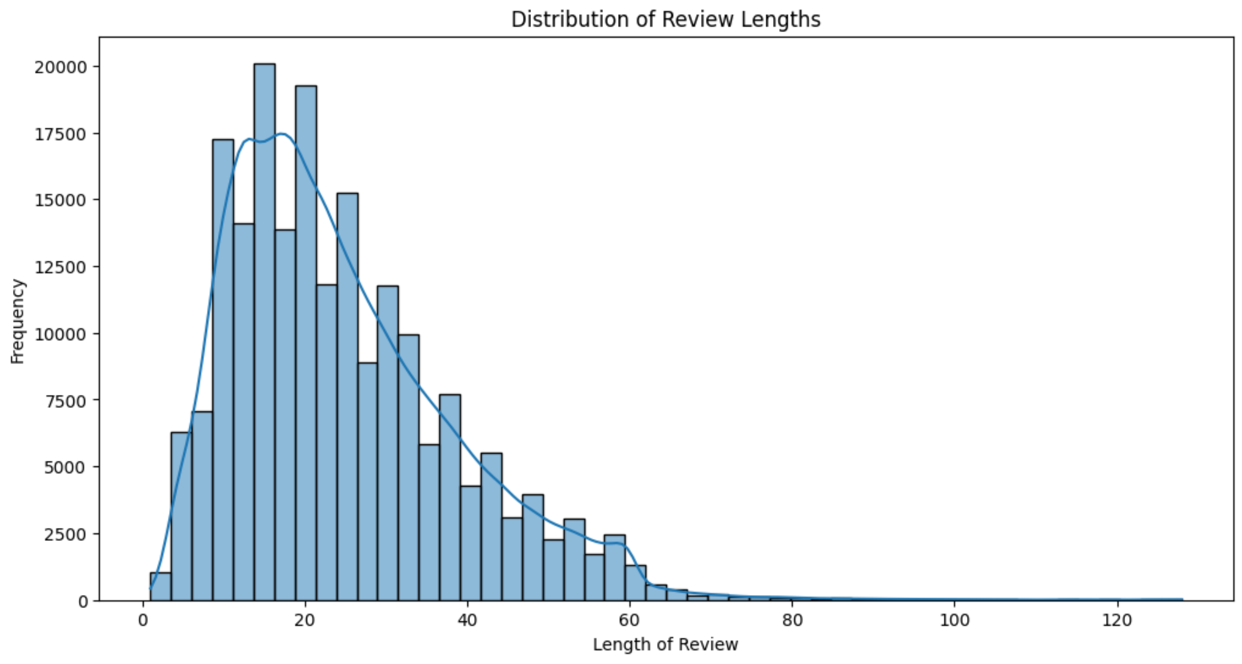


Figure 1 - Histogram: Review Length Distribution

2.2 Dataset Suitability and Limitations

The Amazon Reviews Polarity dataset was selected for several advantageous properties. Its substantial size, even in the 200,000-review subset used, helps mitigate the risk of model overfitting and supports the training of complex deep learning architectures. The perfectly balanced class distribution simplifies model evaluation, allowing for straightforward interpretation of metrics like accuracy, precision, and recall without concerns about class imbalance bias. Furthermore, the inherent diversity of product domains within the Amazon reviews encourages the development of models that generalize well beyond specific topics or writing styles.

However, the dataset also has limitations. The primary limitation is the restriction to binary sentiment classification (positive/negative), precluding analysis of neutral sentiment or more nuanced emotional states. Additionally, as is common with large, user-generated text corpora, the dataset may contain noise, such as irrelevant content, grammatical errors, duplicated reviews, or inconsistencies that could potentially affect model training, although standard pre-processing steps aim to minimize these effects.

2.3 Alternative Datasets Considered

Several other widely used sentiment analysis datasets were considered during the initial project planning phase. These included the IMDB Movie Reviews dataset, the Yelp Reviews dataset, and the Stanford Sentiment Treebank (SST-2).

Although these datasets represent valuable benchmarks and offer distinct textual characteristics (e.g., movie versus product reviews), they were excluded to maintain a clear experimental focus on evaluating the chosen deep learning architectures specifically on Amazon Reviews Polarity data.

2.4 Our Pre-processing Pipeline

A comprehensive pre-processing pipeline was implemented to transform the raw review text into an optimized form suitable for deep learning models. The following steps were applied sequentially, using Pandas for data manipulation and Keras utilities for text processing:

2.4.1 Data Loading

The dataset was loaded from its CSV format, initially specifying the loading of the first 200,000 rows (NROWS_TO_LOAD = 200000). Column names ('polarity', 'title', 'text') were assigned during loading

2.4.2 Handling Missing Values

Rows containing missing values in either the 'title' or 'text' fields were identified and removed to ensure data integrity. This resulted in a slight reduction from the initial 200,000 rows

2.4.3 Text Combination

The 'title' and 'text' columns were concatenated into a single 'full_review' column for each sample, providing the models with combined context from both fields.

2.4.4 Text Cleaning

A cleaning function was applied to the 'full_review' text. This involved things like converting all text to lowercase, removing characters that were not lowercase letters or whitespace characters, and normalizing whitespace by collapsing multiple spaces, tabs, or newlines into single spaces and removing leading/trailing whitespace.

2.4.5 Stop Word Removal

During the data exploration phase, we visualized the 20 most common words in the dataset. As shown in the frequency distribution (figure 2), these were dominated by common words such as “the", "and", "to”, and “is". Based on this we incorporated stop word removal as a core part of preprocessing.

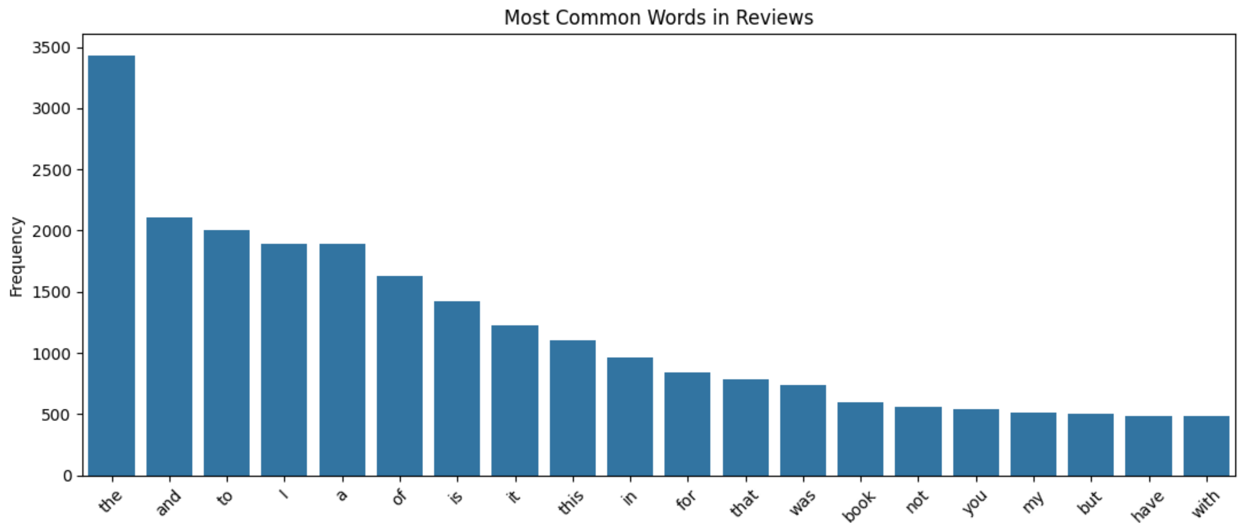


Figure 2 - Most common words in reviews

English stop words were removed from the cleaned text using a standard list (specifically, the stopwords.words('english') list from the NLTK library). This step aimed to reduce noise and focus the model on more informative words. The resulting text was stored in a new column, designated as the final text input for the models.

2.4.6 Label Mapping

The original polarity labels ('1' for negative, '2' for positive) were mapped to a binary format suitable for model training: '0' for negative and '1' for positive.

2.4.7 Data Splitting

The processed dataset (features: cleaned text without stop-words; labels: binary 0/1) was split into three distinct sets: training (60%), validation (20%), and testing (20%). This split was performed using scikit-learn's train_test_split function with stratification based on the labels (stratify=y). Stratification ensures that the proportion of positive and negative samples is approximately preserved across all three sets, which is crucial given the balanced nature of the original data. The split was achieved in two stages: first separating the validation set (20%) and then splitting the remaining 80% into training (75% of remainder, i.e., 60% of total) and test (25% of remainder, i.e., 20% of total).

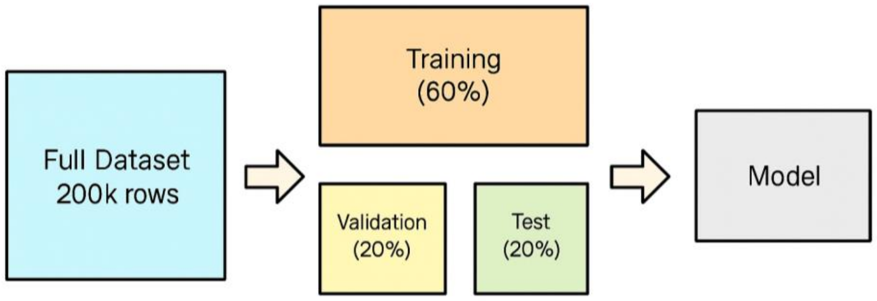


Figure 3 - Data splitting visual diagram (source: authors)

2.4.8 Tokenization

A Keras Tokenizer was initialized, configured with a specific maximum vocabulary size (VOCAB_SIZE) and an out-of-vocabulary token (OOV_TOKEN) to handle words not present in the vocabulary learned from the training data. Crucially, the tokenizer was fitted *only* on the text data from the *training set* (X_train) to prevent data leakage from the validation or test sets into the vocabulary creation process.

2.4.9, Sequence Conversion

The fitted tokenizer was then used to convert the text data from the training, validation, and test sets into sequences of integer indices (texts_to_sequences). Each word in a review was replaced by its corresponding integer index from the tokenizer's vocabulary

2.4.10 Padding and Truncation

To feed the network a uniformly shaped tensor, every review was either padded or truncated to a fixed length. The cut-off was selected only from the *training* split to avoid information leakage: the length histogram shows a long-tailed distribution with a median of

≈ 37 tokens, a mean of ≈ 41 tokens, and roughly 93 % of all reviews at or below 80 tokens. Based on that, two limits were adopted. For the CNN and the GRU model the maximum sequence length was set to 70 tokens, retaining about the 90-th percentile while keeping memory demand low; the bidirectional LSTM, which can benefit from slightly longer context, was given a limit of 80 tokens, still below the 95-th percentile. Sequences shorter than the chosen limit are padded at the end (padding="post"), while longer sequences are truncated from the tail (truncating="post"). Padding uses the value 0, and the embedding layer is configured with mask_zero=True, so the networks learn to ignore these filler tokens. The fitted Keras Tokenizer is saved to disk with pickle, ensuring that the identical word-index mapping is available whenever the model is evaluated or deployed later.

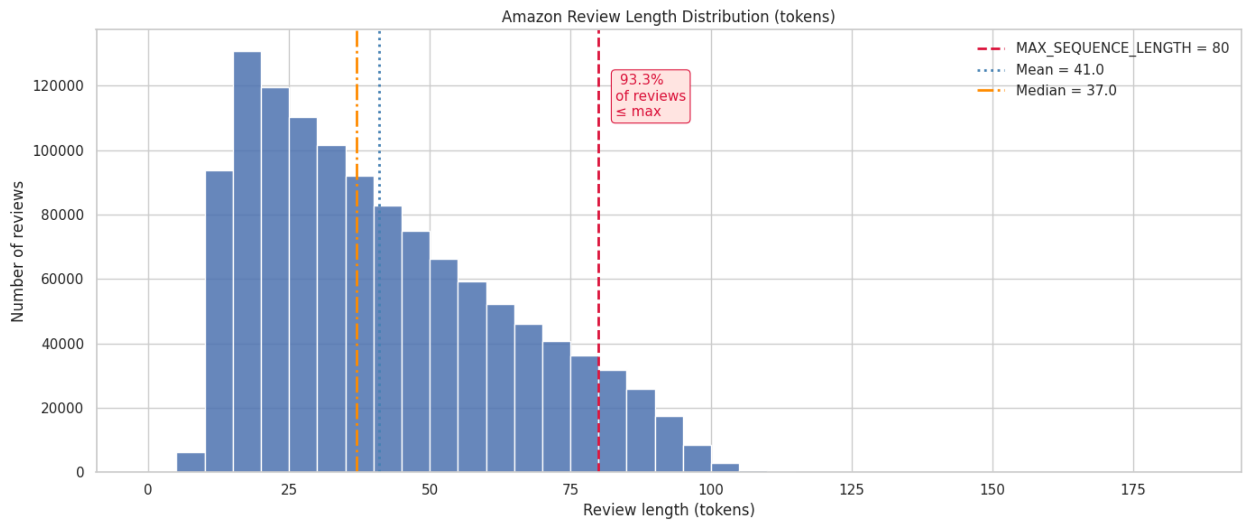


Figure 4 - Review Length & Frequency

3 Model Implementation

This section details the deep learning architectures implemented for the sentiment classification task. We describe the specific models chosen, the theoretical motivations behind their selection, their key hyperparameters and optimization strategies, and the procedure followed for training.

3.1 Architectures Considered

The core of this project involved the implementation and comparison of several neural network architectures for text classification. A standard 1D Convolutional Neural Network (CNN) represented a common non-sequential approach frequently cited in literature. Additionally, two recurrent neural network (RNN) architectures known for their effectiveness in sequence modeling tasks, Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU), were implemented. Fine-tuning a pre-trained Transformer model like DistilBERT was also considered as a benchmark to represent the approximate upper bound performance achievable with large-scale language models on this task, although the focus remained on training the CNN and RNN models from scratch. The primary experimental goal was therefore to evaluate and contrast the performance of the CNN, BiLSTM and BiGRU models against each other.

Model	***Type***	***Sequence Handling***	***Trainable Params***	***Strengths***
CNN	Non-Sequential	No	~1.2m	Fast training, good at capturing local patterns, highly parallelizable
BiGRU	Sequential	BiDirectional	~690k	Efficient with fewer parameters, captures sequential context, faster training than LSTM
BiLSTM	Sequential	BiDirectional	~1.8m	Excellent for long-range dependencies, rich temporal context modeling

Table 1 - Model comparison table

3.2 Theoretical Motivation

The CNN model represented a different approach to text classification compared to the RNNs. CNNs operate by applying convolutional filters across the sequence embeddings, enabling them to learn local n-gram features indicative of sentiment (Yoon, 2014). These filters act as feature detectors that scan the text and identify patterns irrespective of their position in the sequence (Jacovi, Shalom, & Goldberg, 2018). Max-pooling layers reduce dimensionality and allow for translational invariance, while dropout layers help prevent overfitting. CNNs are highly parallelizable, which makes them fast to train even on large datasets, and suitable for real-time applications. While not inherently designed for sequential order processing in the same way as RNNs, CNNs have proven effective for text classification tasks by efficiently identifying key phrases or word combinations that are strongly indicative of sentiment. Because the decisive cues in review language are usually short phrases (for example, “works perfectly” or “waste of money”), a window-based technique can capture enough context without a recurrent state, provided that multiple filter widths are stacked, and global pooling is applied.

In contrast, RNNs are designed to handle sequential data and capture temporal dependencies. Both the GRU and LSTM models were implemented with a bidirectional wrapper. This means the models process the input text in both forward (left-to-right) and backward (right-to-left) directions. The outputs from both directions are then concatenated before being passed to the next layer. This architectural choice provides the neural network context from both earlier and later parts of a sentence when interpreting each word (Schmidhuber & Graves, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, 2005). Such contextual awareness is particularly important for correctly interpreting sentiment when the phrasing involves complex phrasing, negation, or contrast, for instance, in sentences like "This product *would have been* great *if it actually worked*."

A central comparison in this project was between the LSTM and GRU recurrent cells. LSTMs incorporate three distinct gates (input, forget, output) alongside a separate cell state (Schmidhuber & Hochreiter, 1997). This complex structure potentially allows them to capture longer-range dependencies and regulate information flow more effectively within the sequence, though it results in a higher number of trainable parameters. In contrast, GRUs utilize a simpler architecture featuring only two gates (reset and update) and combines the cell state and hidden state (Bengio, et al., 2014). This design leads to fewer parameters compared to LSTMs, which can translate to faster training times and a potentially reduced risk of overfitting, particularly with limited data, while often achieving comparable performance on many sequence modeling tasks (Bengio, Cho, Gulcehre, & Chung, 2014) . Evaluating this trade-off between complexity and performance was a key aspect of the investigation and is mentioned in 5.4. Retaining the BiLSTM therefore lets us quantify whether the extra gate and its 1.1 million additional parameters secures any measurable edge on this dataset, information that guides future model selection for deployments where memory or latency is critical.

3.3 Model Architectures, Hyperparameters, and Optimization

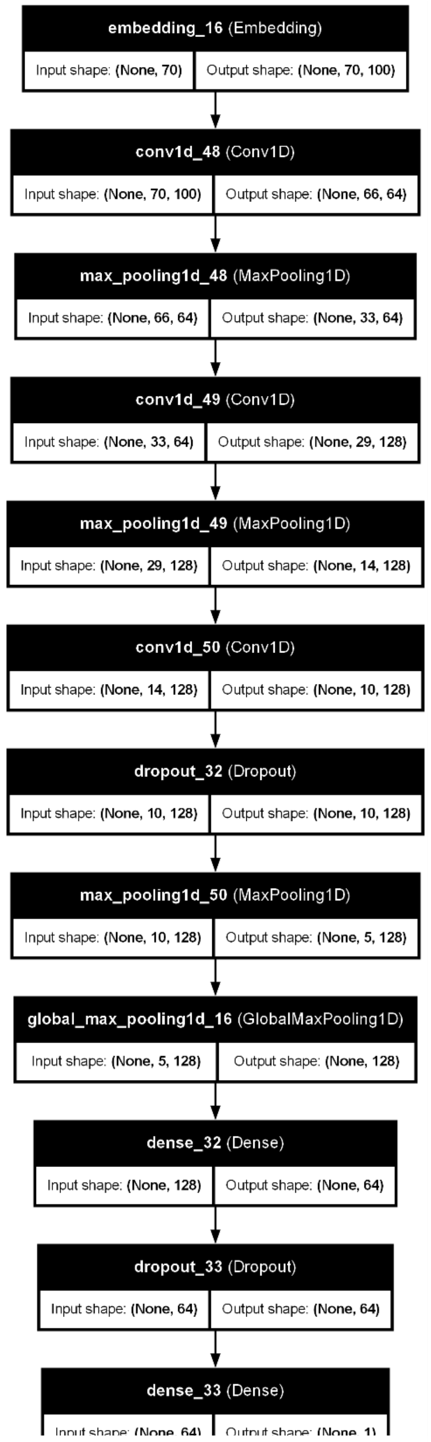

embedding_16 (Embedding)
Input shape: (None, 70)
Output shape: (None, 70, 100)
conv1d_48 (Conv1D)
Input shape: (None, 70, 100)
Output shape: (None, 66, 64)
max_pooling1d_48 (MaxPooling1D)
Input shape: (None, 66, 64)
Output shape: (None, 33, 64)
conv1d_49 (Conv1D)
Input shape: (None, 33, 64)
Output shape: (None, 29, 128)
max_pooling1d_49 (MaxPooling1D)
Input shape: (None, 29, 128)
Output shape: (None, 14, 128)
conv1d_50 (Conv1D)
Input shape: (None, 14, 128)
Output shape: (None, 10, 128)
dropout_32 (Dropout)
Input shape: (None, 10, 128)
Output shape: (None, 10, 128)
max_pooling1d_50 (MaxPooling1D)
Input shape: (None, 10, 128)
Output shape: (None, 5, 128)
global_max_pooling1d_16 (GlobalMaxPooling1D)
Input shape: (None, 5, 128)
Output shape: (None, 128)
dense_32 (Dense)
Input shape: (None, 128)
Output shape: (None, 64)
dropout_33 (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
dense_33 (Dense)
Input shape: (None, 64)
Output shape: (None, 1)

Figure 5 - CNN architecture diagram

The models were constructed using the TensorFlow Keras API. Specific architectural choices and key hyperparameters were established based on initial data exploration, such as analyzing sequence length distributions, and refined through manual tuning guided by performance on the validation set. The Convolutional Neural Network (CNN) started with an Embedding layer using a vocabulary of 10,000 words and 100-dimensional embeddings for sequences of up to 70 tokens. The core of the CNN consisted of a sequence of three Conv1D layers with 'relu' activation and a kernel size of 5. The filter sizes were 64, 128, and 128, respectively. Each Conv1D layer was followed by a MaxPooling1D layer with a pool size of 2. A Dropout layer with a rate of 0.3 was added after the third convolutional block. Feature aggregation was performed using a GlobalMaxPooling1D layer. Finally, a Dense layer with 64 ReLU units is followed by another Dropout layer (rate 0.3). The final Dense layer uses a sigmoid activation function for binary classification. The CNN was optimized using Adam optimizer with a learning rate of 1e-3 and binary_crossentropy loss.

The Bidirectional GRU (BiGRU) model uses an Embedding layer for a 10,000-word vocabulary, generating 64-dimensional embeddings. It processes input sequences of 70 tokens and includes masking. The subsequent SpatialDropout1D layer had a rate of 0.3. The Bidirectional (GRU) layer contained GRU cells with 64 units, an input dropout of 0.3, and recurrent dropout set to 0.0, also configured with return_sequences=False. The output layer is a single Dense neuron with sigmoid activation. For optimization, the Adam optimizer was used with a learning rate of 1e-3 and the binary_crossentropy loss. This model has a total of 690 049 trainable parameters.

The Bidirectional LSTM (BiLSTM) model employed a similar structure but with different hyperparameters. It began with an Embedding layer configured for a vocabulary size of 12,000, producing 128-dimensional vectors for input sequences of 80 tokens, and utilized masking (mask_zero=True) to ignore padding. This was followed by a SpatialDropout1D layer with a rate of 0.4 for regularization. The core layer was a Bidirectional (LSTM) wrapper containing LSTM cells with 128 units, an input dropout rate of 0.4, and recurrent dropout set to 0.0 for GPU optimization (return_sequences=False). The final layer is again a single Dense neuron with a sigmoid activation function for binary classification. This model was optimized using the Adam optimizer with a learning rate of 1e-3 and the binary_crossentropy loss function. The model has a total of 1 799 425 trainable parameters.

Model	Vocab size	Embedding dim	Sequence length	Hidden units	Dropout rate	Learning rate	Optimizer	Activation
CNN	10,000	100	70	-	0.3 (2 layers)	1e-3	Adam	ReLu / Sigmoid
BiGRU	10,000	64	70	64	0.3	1e-3	Adam	Sigmoid
BiLSTM	12,000	128	80	128	0.4	7e-4	Adam	Sigmoid

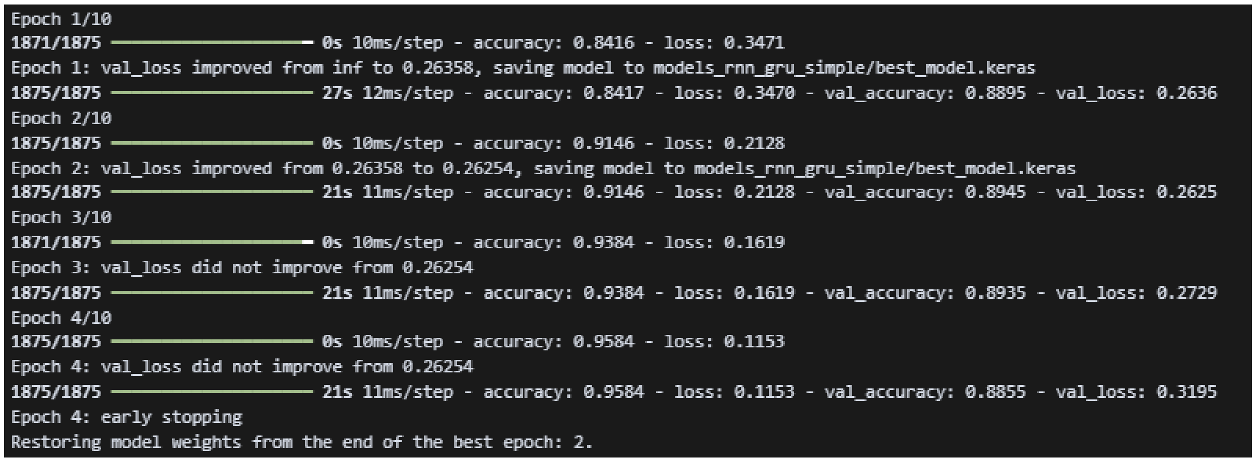
Table 2 - Hyperparameter summary table

3.4 Training Procedure

All implemented models followed a standardized training procedure. The pre-processed data was divided into training (60%), validation (20%), and testing (20%) sets, with stratification to ensure class balance across splits, as detailed in Section 2.4. Models were trained with a batch size of 64, which offered a good trade-off between training stability and memory efficiency.

The Adam optimizer and binary_crossentropy loss function were used for model compilation, and accuracy was monitored as a key performance metric during training runs. To prevent overfitting and to retain the best-performing weights, an EarlyStopping callback mechanism was integrated into the training process. This mechanism monitored the validation loss after each epoch and halted training if the loss did not improve for two consecutive epochs (patience=2). The callback was configured to restore the model weights corresponding to the epoch that yielded the lowest validation loss.

Model training was conducted on different hardware setups. Some training runs were performed on a local PC, while others utilized Google Colab Pro with A100 GPU acceleration. On Colab Pro, typical epoch durations ranged from approximately 23-28 for the RNN models. The other setup used a local PC equipped with Intel® Core™ i7-10700F CPU and an RTX 2080 Super GPU. On this local machine, epoch durations were approximately 71-86 seconds for the RNN models and 24 seconds for the CNN model. While maximum epoch limits were set in the configurations, training often concluded earlier due to the early stopping condition being met.



```
Epoch 1/10
1871/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8416 - loss: 0.3471
Epoch 1: val_loss improved from inf to 0.26358, saving model to models_rnn_gru_simple/best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 27s 12ms/step - accuracy: 0.8417 - loss: 0.3470 - val_accuracy: 0.8895 - val_loss: 0.2636
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9146 - loss: 0.2128
Epoch 2: val_loss improved from 0.26358 to 0.26254, saving model to models_rnn_gru_simple/best_model.keras
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.9146 - loss: 0.2128 - val_accuracy: 0.8945 - val_loss: 0.2625
Epoch 3/10
1871/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9384 - loss: 0.1619
Epoch 3: val_loss did not improve from 0.26254
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.9384 - loss: 0.1619 - val_accuracy: 0.8935 - val_loss: 0.2729
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9584 - loss: 0.1153
Epoch 4: val_loss did not improve from 0.26254
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 11ms/step - accuracy: 0.9584 - loss: 0.1153 - val_accuracy: 0.8855 - val_loss: 0.3195
Epoch 4: early stopping
Restoring model weights from the end of the best epoch: 2.
```

Figure 6 - Console snippet showing GRU model training and EarlyStopping

4 Evaluation

This section presents the evaluation of the trained sentiment analysis models. We outline the metrics used for assessment, compare the performance of the CNN, Bidirectional GRU and Bidirectional LSTM models on the held-out test set, analyze training dynamics, architectural efficiency, and briefly introduce model explainability.

4.1 Evaluation Metrics

To comprehensively assess model performance, a standard set of classification metrics was employed. Accuracy provides an overall measure of correct predictions, calculated as the ratio of correctly classified reviews to the total number of reviews in the test set. Given the dataset's balanced nature (50% positive, 50% negative), the majority-class baseline accuracy is 50%, providing a simple benchmark against which model performance can be judged. Precision, recall, and the F1-score were calculated for each class (positive and negative) to provide deeper insights into the model's ability to correctly identify instances of a specific class and avoid misclassifications. Precision measures the proportion of predicted positive (or negative) reviews that were correct (or incorrect), while recall measures the proportion of actual positive (or negative) reviews that were correctly identified. The F1-score represents the harmonic mean of precision and recall, offering a single metric that balances both concerns.

Confusion matrices were generated to show the absolute counts of true positives, true negatives, false positives, and false negatives. Finally, the Receiver Operating Characteristic (ROC) curve was plotted, and the Area Under the Curve (ROC-AUC) was calculated. The ROC-AUC score quantifies the model's ability to discriminate between the positive and negative classes across different classification thresholds, with a value of 1.0 indicating perfect separation and 0.5 indicating performance equivalent to random guessing.

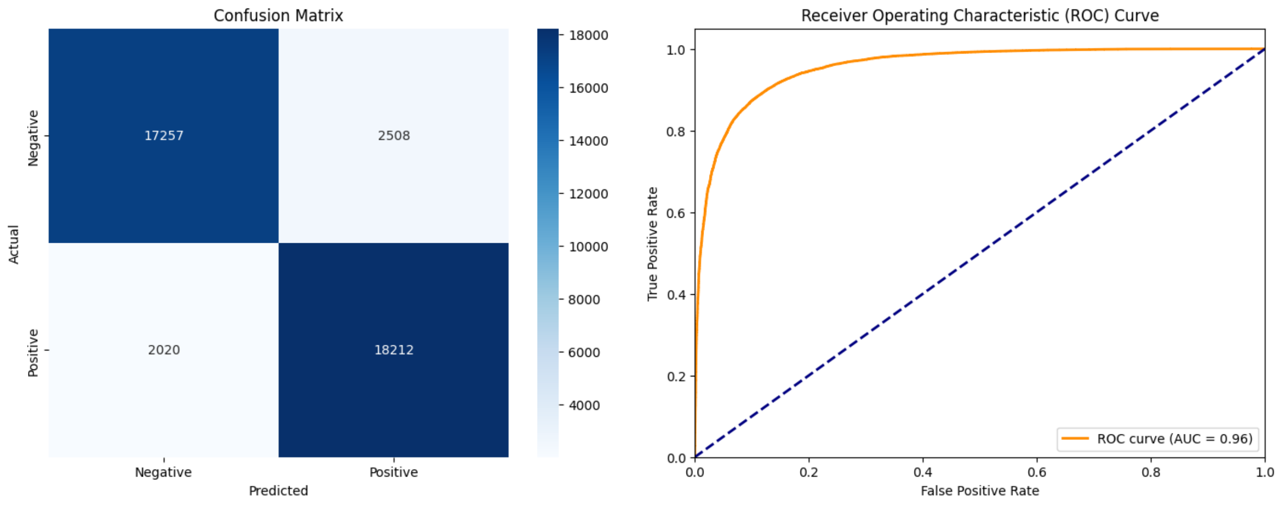


Figure 7 - Confusion Matrix and ROC-AUC plots for the CNN model

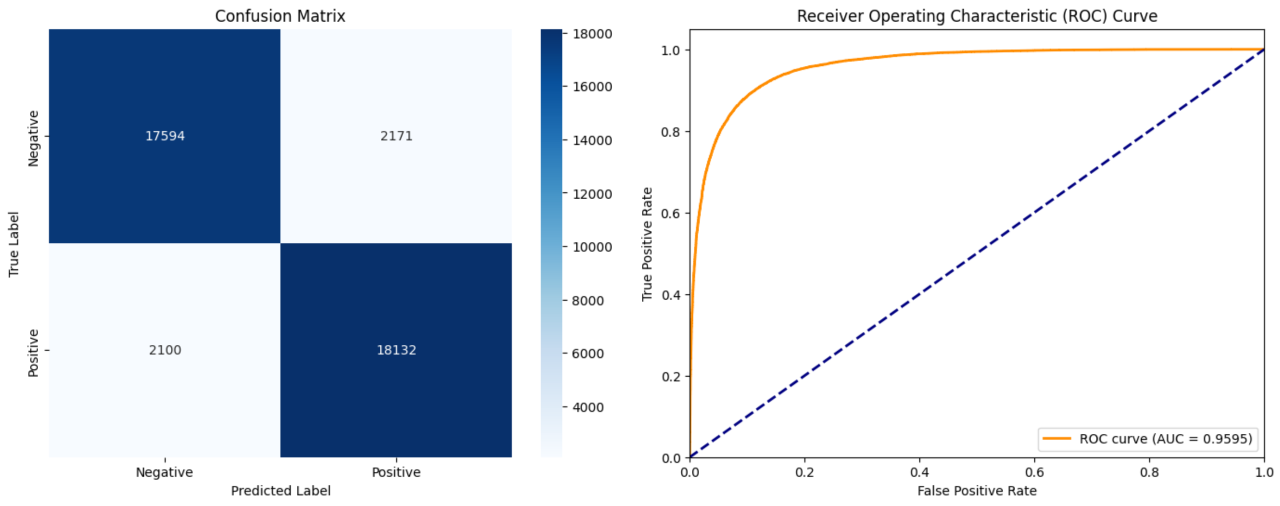


Figure 8 - Confusion Matrix and ROC-AUC plots for the GRU model

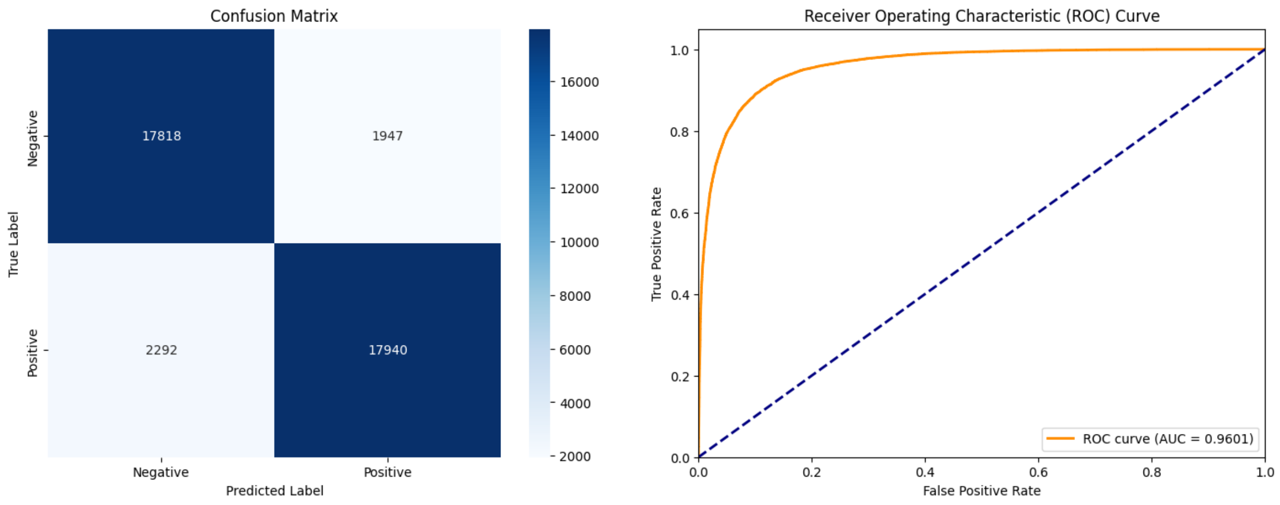


Figure 9 - Confusion Matrix and ROC-AUC plots for the LSTM model

4.2 Model Performance Comparison

The trained models were evaluated on the held-out test set using the metrics described in the previous section. The Bidirectional GRU and Bidirectional LSTM models achieved nearly identical overall performance, both reaching an accuracy of 90%. The CNN model performed slightly lower, achieving an accuracy of 89%.

Examining the class-specific metrics for the BiGRU model, the precision, recall, and F1-score were consistently around 0.89-0.90 for both the negative and positive classes, indicating balanced performance. Its confusion matrix revealed 17,594 true negatives, 2,171 false positives, 2,100 false negatives, and 18,132 true positives. The BiGRU model achieved a strong ROC-AUC score of 0.9595.

The BiLSTM model also demonstrated balanced performance with precision, recall, and F1-scores around 0.89-0.91 across both classes, resulting in macro and weighted averages of 0.90 for these metrics, matching the overall accuracy. Its confusion matrix showed 17,818 true negatives, 1,947 false positives, 2,292 false negatives, and 17,940 true positives (note: evaluated on a slightly different test set split configuration compared to GRU/CNN). The BiLSTM achieved a marginally higher ROC-AUC score of 0.9601, suggesting a very slightly superior, yet practically equivalent, discriminative ability compared to the BiGRU.

The CNN, with its 89% accuracy, showed slightly less balanced precision and recall compared to the RNNs. For the negative class (0), it achieved 0.90 precision but 0.87 recall, while for the positive class (1), it had 0.88 precision and 0.90 recall. The resulting F1-scores were 0.88 and 0.89, respectively. The confusion matrix for the CNN indicated 17,257 true negatives, 2,508 false positives, 2,020 false negatives, and 18,212 true positives. Despite the slightly lower accuracy, the CNN achieved a ROC-AUC score of 0.96, demonstrating a comparably strong ability to distinguish between positive and negative reviews overall, similar to the RNN models. In summary, both the BiGRU and BiLSTM models delivered the target performance of approximately 90% accuracy, slightly outperforming the CNN (89%)

on this task. All three models exhibited excellent discriminative capability as evidenced by their high ROC-AUC scores around 0.96.

Model	***Accuracy***	***Precision (avg)***	***Recall (avg)***	***F1-score (avg)***	***ROC-AUC***
CNN	89%	0.88/0.90	0.87/0.90	0.88/0.89	0.96
BiGRU	90%	0.90/0.89	0.89/0.91	0.89/0.90	0.9595
BiLSTM	90%	0.90/0.89	0.88/0.91	0.89/0.90	0.9601

Table 3 - Table comparing Accuracy, Precision, Recall, F1 and ROC-AUC across all models – The slash indicates positive/negative

4.3 Training Dynamics and Early Stopping

The training logs for all three models revealed a common pattern. Performance on the training data generally improved rapidly over the initial epochs, as indicated by increasing accuracy and decreasing loss. However, performance on the validation set tended to plateau relatively quickly. For both the BiGRU and BiLSTM models, the lowest validation loss (indicating the best generalization performance before potential overfitting) was achieved at epoch 2. The early stopping mechanism, configured with a patience of 2, subsequently halted training after epoch 4 for both models when validation loss failed to improve further. The models' weights were automatically restored to the state captured at the end of epoch 2. A similar dynamic was observed for the CNN model, where validation accuracy peaked early (epoch 2) and validation loss began to increase thereafter, suggesting that optimal generalization was achieved within the first few epochs of training. This consistent early stopping highlights the models' ability to learn effective representations relatively quickly on this dataset and the importance of validation-based stopping to prevent overfitting.

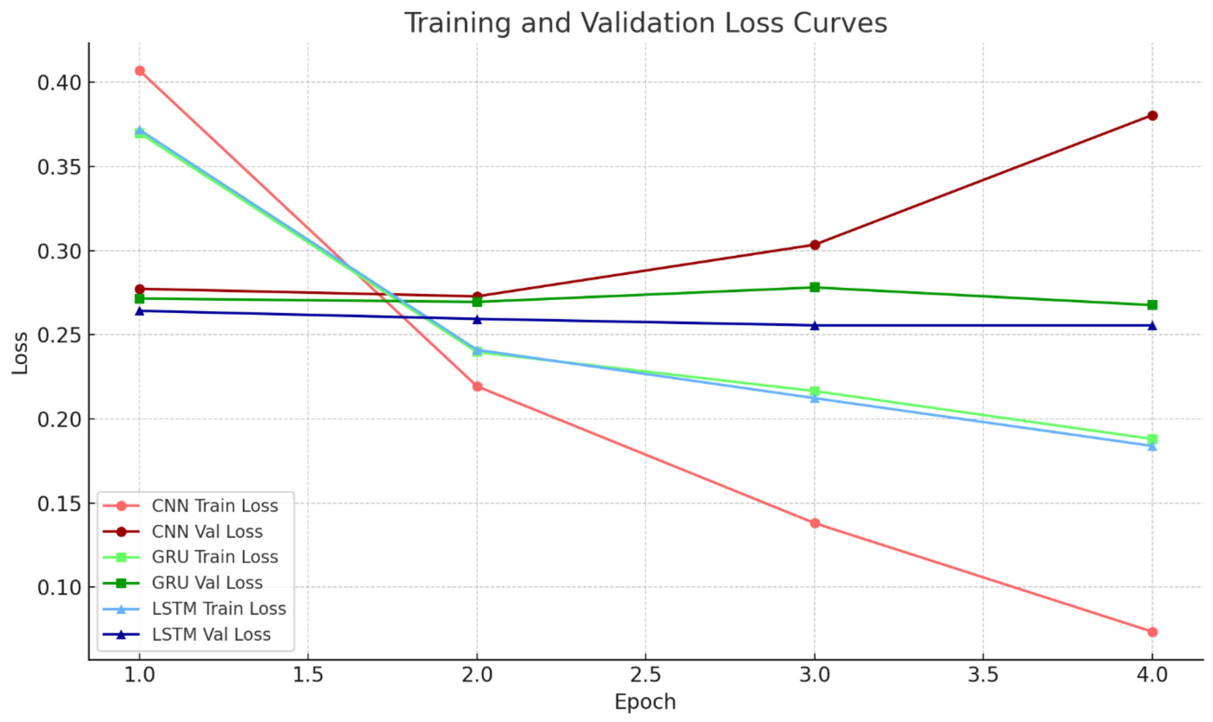


Figure 10 - Training and validation loss for all models

4.4 Architecture and Efficiency Considerations

During the implementation of the RNN models (BiGRU and BiLSTM), we made several architectural choices to improve computational efficiency without sacrificing performance. Because NVIDIA's CuDNN kernels do not support recurrent_dropout, we set recurrent_dropout = 0.0 in both models. For the BiGRU we could still use the CuDNN-friendly dropout argument (input-dropout) inside the layer, whereas the BiLSTM retained CuDNN speed by keeping its internal dropout at zero and relying solely on an external SpatialDropout1D applied to the embeddings for regularisation. These measures delivered the expected ~3x speed-up with CuDNN while still guarding against overfitting.

4.5 Explainability Approach

While the primary evaluation relied on quantitative metrics, qualitative understanding of model decisions is also important. LIME (Local Interpretable Model-agnostic Explanations) was used to analyze model predictions and identify influential tokens (Kulkarni, 2024).

LIME creates interpretable local surrogate models to explain individual predictions. In other words, it explains why the model made a specific prediction by using a simpler model that's easy to understand (Ribeiro, Singh, & Guestrin, 2016). This was particularly helpful in visualizing which words or phrases contributed most strongly to specific sentiment classifications. For example, terms like “refund” and “disappointed” were often associated with negative predictions, while words such as “excellent” and “love” supported positive predictions. These observations aligned well with human intuition and helped to validate the fact that the models were not relying on spurious correlations.

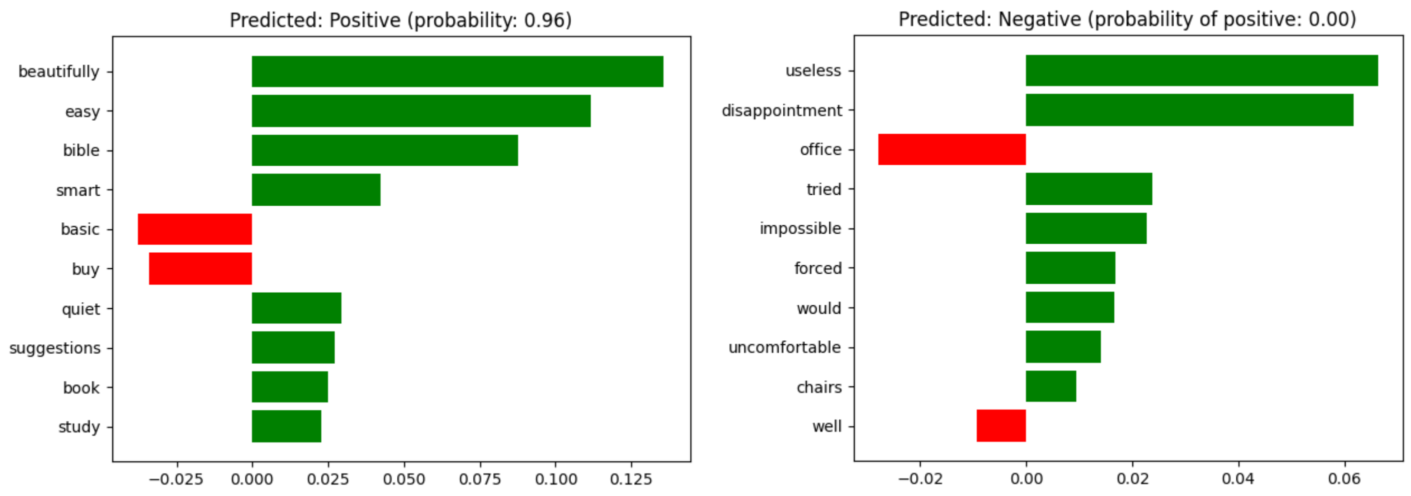


Figure 11 - Lime visualizations highlighting influential words for sample predictions

5 Results and Discussion

This section discusses the implications of the evaluation results presented in Section 4. We compare the performance of the implemented models, delve into potential sources of error, consider the impact of design choices like sequence length, situate our models within the broader context of model size versus accuracy, and suggest avenues for future work.

5.1 Quantitative Comparison

The evaluation results demonstrate that both the Bidirectional LSTM (BiLSTM) and Bidirectional GRU (BiGRU) models successfully achieved the target performance level, reaching 90% accuracy on the held-out test set. This represents a substantial improvement over the 50% random baseline and aligns with the performance of established benchmarks like FastText on similar tasks (Surve, Momin, Sawadkar, & Muskawad, 2020). When comparing the two primary RNN architectures, their performance was remarkably similar across all major metrics. Precision and recall were well-balanced at approximately 0.90 for both positive and negative classes in both models, indicating no significant bias towards one class over the other. While the BiLSTM achieved a marginally higher ROC-AUC score (0.9601) compared to the BiGRU (0.9595), this difference is minimal and likely falls within the range of statistical noise, suggesting no practical advantage in discriminative capability for the LSTM on this specific dataset and task configuration. The CNN performed slightly below the RNNs with 89% accuracy, potentially reflecting the RNNs' enhanced ability to capture sequential dependencies crucial for nuanced sentiment, although its ROC-AUC of 0.96 indicates its fundamental feature extraction was still highly effective. Overall, the results suggest that for this task, the simpler BiGRU architecture offers comparable performance to the more complex BiLSTM.

5.2 Error Analysis

Despite the high overall accuracy, an inspection of potential misclassifications reveals two common challenges inherent in sentiment analysis. False negatives, where negative reviews are incorrectly classified as positive, may often occur with sarcastic text. Sarcasm frequently uses positive language to convey negative sentiment (e.g., "Absolutely amazing how quickly this broke"). Models relying heavily on surface-level word polarity can miss this cue. False-positives, on the other hand, arise when an otherwise negative review contains a few strongly positive phrases, typically referring to an unrelated aspect like “great packaging” or “fast delivery”. These phrases influence the model’s final prediction, overshadowing the core sentiment about the product itself.

BiDirectional context was used on the RNN models to combat these errors. In the CNN we removed high-frequency stop words and used a narrower max-pool window, which helps the filters ignore empty intensifiers (“absolutely”, “really”). Addressing these error patterns often requires richer context modelling and a deeper grasp of discourse.

5.3 Effect of Sequence Length

The choice of maximum sequence length (80 tokens for LSTM, 70 for GRU/CNN) was informed by dataset characteristics, where only about 10% of reviews exceeded 80 tokens after preprocessing. This implies that truncation affected only a small minority of the input samples. The high performance achieved shows that this truncation did not lead to significant information loss detrimental to sentiment classification. This confirms that for most of the reviews in the dataset, the primary sentiment indicators tend to appear relatively early in the text, and extending the sequence length further offered diminishing returns, validating the chosen lengths as a reasonable balance between information capture and computational efficiency.

5.4 Model Size vs. Accuracy Trade-off

A key goal of this project was to achieve strong performance with relatively lightweight models. The implemented Bi-LSTM model, with approximately 1.8 million trainable parameters, achieved 90% accuracy. The same was true for the Bi-GRU model, which had only approx. 690k parameters. This performance is notably lower than state-of-the-art results reported in the literature using large pre-trained Transformer models like BERT, which often exceed 95% accuracy on this dataset. However, these large models typically have over 100 million parameters (e.g., BERT-base has ~110M), making our Bi-LSTM and Bi-GRU models ~ 61x and ~159x smaller, respectively (Devlin, Chang, Lee, & Toutanova, 2019). This significant difference in parameter size translates directly to reduced training time, lower memory requirements, and faster inference speeds. Therefore, while sacrificing ~ 3-5 percentage points in accuracy compared to the state-of-the-art, our RNN models represent a pragmatic middle ground, offering substantial sentiment analysis capabilities within a computationally efficient package suitable for deployment scenarios where resource constraints are a key consideration.

5.5 Impact of scaling the dataset for performance

Although this report is mainly written about how our models performed when trained on a ~ 200k subset of the full dataset, we were curious on what would happen if we simply increased the amount of data we used to train, validate and test the models with, without changing anything else. To do this we simply ran extra training runs for each model (CNN, BiGRU & BiLSTM) with “NROWS_TO_LOAD” from 200,000 up to 2 million. A familiar but important result: the more relevant data we threw at our models, the better they performed. This was true for all metrics, and for overfitting, as one can see in the “Training and Validation Loss” plots further down. To quantify these results, we have created a graph that tracks some metric across the different dataset sizes. We were originally going to show how the test-loss decreased as we increased the dataset, but we sadly forgot to save this during the runs, and we didn’t have the time to rerun the models at that dataset size when we noticed this. Because of this, we have chosen to show how the test-accuracy is our improved metric.

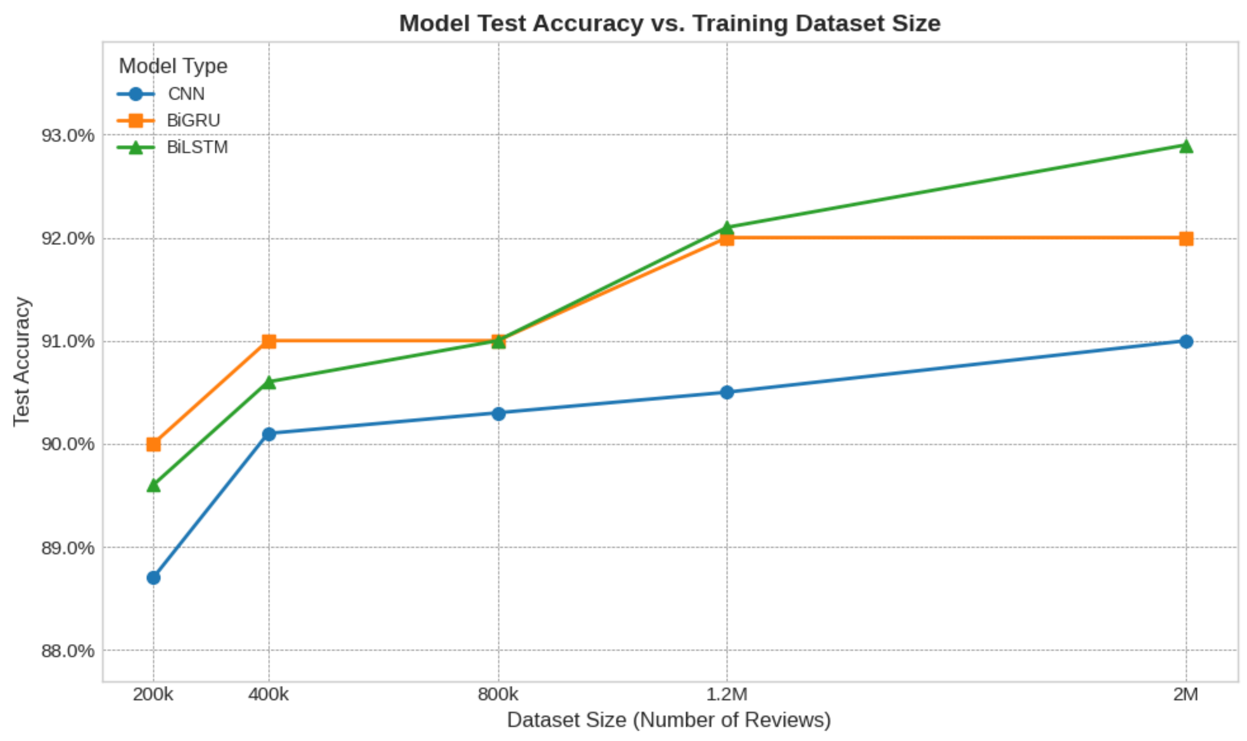


Figure 12 – Test accuracy with increasing dataset size

As depicted in figure 12, all three architectures demonstrate a consistent improvement in test accuracy as the number of training reviews increases from 200,000 to 2 million. This empirically validates the well-established principle that deep learning models, particularly for complex tasks like sentiment analysis, benefit significantly from larger datasets, allowing them to learn more robust and generalizable patterns (Hestness, et al., 2017) (Kaplan, et al., 2020). The “godfather” of RL, Richard Sutton, is known for sharing this view. In his 2019 essay “The Bitter Lesson” he speaks about the fact that AI methods which leverages computation, which in order leverages data to learn search and planning, have historically always beaten methods that rely on specific hardcoded human knowledge, to solve problems in AI (Sutton, 2019).

Specifically, the CNN model's accuracy increased from approximately 88.7% with 200k reviews to 91.0% with 2M reviews. The BiGRU model improved from 90.0% to 92.0%, while the BiLSTM showed the largest overall gain, rising from 89.6% to a peak of 92.9% accuracy at the 2 million review mark.

Interestingly, the rate of improvement varies. The most substantial gains for all models are observed when increasing the dataset size from 200k up to around 800k or 1.2M. Beyond this point, the curves begin to flatten, particularly for the BiGRU model, which sees no improvement between 1.2M and 2M reviews, and the CNN, which shows only modest gains. This suggests potential diminishing returns, where doubling the data yields progressively smaller accuracy increases. The BiLSTM model, however, appears to benefit more consistently from the largest dataset size compared to the other two.

Furthermore, the graph reinforces the findings from the primary 200k evaluation: the sequential RNN models (BiGRU and BiLSTM) consistently outperform the CNN, especially as more data becomes available. While the BiGRU initially matches or slightly exceeds the BiLSTM at smaller scales, the BiLSTM architecture ultimately achieves the highest performance when trained on the full 2 million review subset used in these scaling experiments.

In addition to the improvements in final test accuracy, analyzing the training dynamics across the different dataset sizes provides further insight. Below in figure 14 and 15, we show how the “Train vs. Val” loss changed as we added more reviews to the dataset for the Bi-LSTM model.

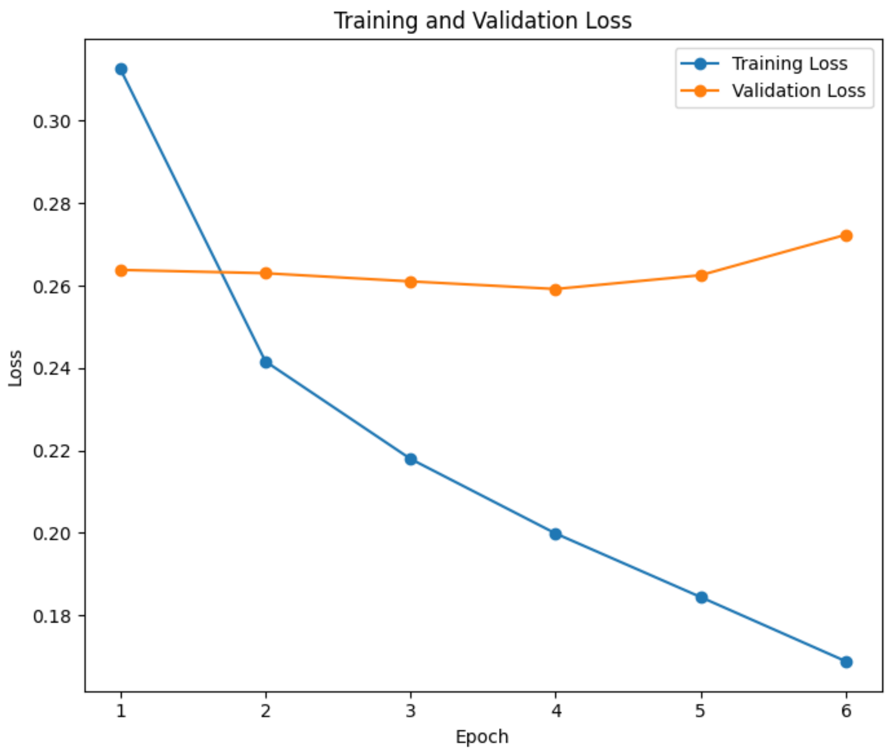


Figure 13 - Training and Validation loss for Bi-LSTM, NROWS = 200k

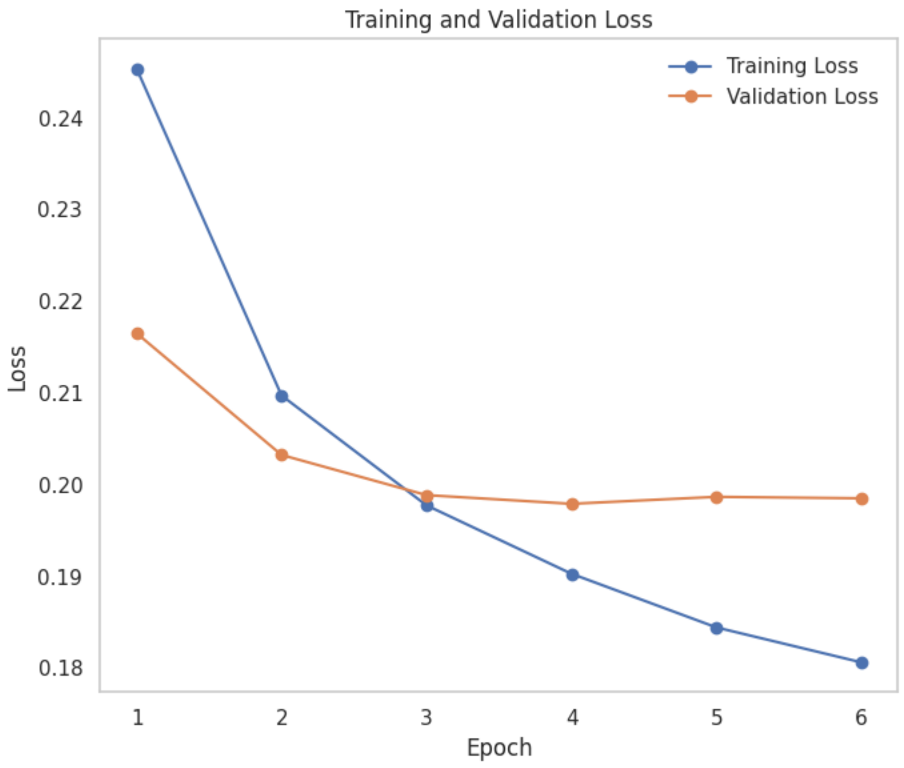


Figure 14 - Training and Validation loss for Bi-LSTM, NROWS = 2000k

A comparison of the training and validation loss curves reveals that the gap between the two curves tends to narrow as the dataset size increases. This indicates a reduction in overfitting; with more data, the models learn patterns that are more representative of the underlying distribution, leading to better generalization as reflected in the validation loss more closely tracking the training loss. This improved generalization during training directly contributes to the higher test accuracies observed with larger datasets and was observed across all models.

5.6 Future Improvements

The project leaves room for strategic enhancements, each capable of elevating the performance and capabilities of the models developed in this project. Employing sub-word tokenization techniques, such as Byte Pair Encoding (BPE) or WordPiece (used by BERT), instead of word-level tokenization, could improve the handling of rare or out-of-vocabulary words and potentially capture morphological variations more effectively. Incorporating attention mechanisms, either as pooling strategies (attention pooling) or as distinct layers within the architecture, could allow the models to dynamically weigh the importance of different parts of the input sequence when making a prediction, potentially improving focus on salient sentiment indicators. Furthermore, knowledge distillation presents an interesting possibility: training our smaller LSTM or GRU model (the "student") to mimic the output probabilities of a larger, pre-trained model like BERT (the "teacher") could transfer some of the teacher's capabilities, potentially boosting accuracy without increasing the student model's size or inference cost. Beyond architectural improvements, extending the task's scope, such as introducing a third class to detect spam or fake reviews alongside positive and negative sentiment, represents a valuable functional enhancement, addressing a critical need for maintaining trust in online review platforms.

6 Conclusion

This project set out to develop, evaluate, and compare CNN, BiGRU, and BiLSTM architectures for the task of binary sentiment classification on a large dataset of Amazon product reviews. The goal was to achieve high accuracy comparable to established benchmarks while maintaining computational efficiency suitable for practical deployment scenarios.

The investigation successfully demonstrated the effectiveness of the implemented deep learning models. Both the BiGRU and BiLSTM architectures achieved a strong performance level on the primary 200,000-review subset, reaching 90% test accuracy and exhibiting balanced precision, recall, and F1-scores around 0.90. Their discriminative capabilities were further confirmed by high ROC-AUC scores (approximately 0.96), indicating a robust ability to distinguish between positive and negative sentiment. Notably, the performance difference between the BiGRU and BiLSTM was found to be negligible for this task, suggesting that the structurally simpler GRU offers a compelling balance of performance and efficiency. The CNN also achieved good results (89% accuracy, 0.96 ROC-AUC), reinforcing the viability of deep learning for this task, though slightly trailing the sequence-focused RNNs.

A key contribution of this work lies in demonstrating that these relatively lightweight RNN models, approximately 61x (BiLSTM) to 160x (BiGRU) smaller than large-scale Transformer architectures like BERT, can attain highly competitive performance levels. Furthermore, experiments scaling the training data up to 2 million reviews showed significant accuracy improvements (up to 92.9% for BiLSTM), confirming the models' ability to leverage larger datasets effectively while also exhibiting reduced overfitting. The use of comprehensive evaluation metrics on a large, balanced dataset allowed for a thorough assessment, confirming deep learning as a highly effective approach for automated sentiment analysis.

In conclusion, the project successfully developed and validated efficient deep learning models for sentiment analysis, achieving strong results with architectures significantly more compact than current state-of-the-art alternatives. This provides a solid foundation and demonstrates a pragmatic solution for understanding customer opinions at scale, particularly in resource-constrained environments.

.

Bibliography

Bengio, Y., Bahdanau, D., Cho, K., Merrienboer, B. v., Gulcehre, C., Bougares, F., & Schwenk, H. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv*.

Bengio, Y., Cho, K., Gulcehre, C., & Chung, J. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*.

Chen, T., Samaranayake, P., Cen, X., Qi, M., & Lan, Y.-C. (2022). The Impact of Online Reviews on Consumers' Purchasing Decisions: Evidence From an Eye-Tracking Study. *Frontiers in Psychology*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, 1*, 4171-4186. doi:10.48550/arXiv.1810.04805

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., . . . Zhou, Y. (2017). Deep Learning Scaling is Predictable, Emperically. *arXiv*.

Jacovi, A., Shalom, O. S., & Goldberg, Y. (2018). *Understanding Convolutional Neural Networks for Text Classification*. Retrieved from https://arxiv.org/abs/1809.08037

Jain, K. (2025, 04 22). *Amazon reviews*. Retrieved from Kaggle: https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews/data?select=train.csv

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. *arXiv*, 3.

Kaplan, J., McCandlish, S., Amodei, D., Radford, A., Gray, S., Henighan, T., . . . Child, R. (2020). Scaling Laws for Neural Language Models. *arXiv*.

Kulkarni, A. (2024, 26 02). *LIME for Explainability in Python*. Retrieved 04 24, 2025, from Kaggle: https://www.kaggle.com/code/amitvkulkarni/lime-for-explainability-in-python

Nicely, M. (2024, 05 24). *Accelerating Transformers with NVIDIA cuDNN 9*. Retrieved from Nvidia Developer: https://developer.nvidia.com/blog/accelerating-transformers-with-nvidia-cudnn-9/

Nkhata, G., Zhan, J., & Anjum, U. (2025). Sentiment Analysis of Movie Reviews Using BERT. *arXiv*.

Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. (2011). Finding Deceptive Opinion Spam by Any Stretch of the Imagination. Retrieved from @inproceedings{Ott2011FindingDO,

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrivial* .

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135-1144.

Schmidhuber, J., & Graves, A. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *PubMed*.

Schmidhuber, J., & Hochreiter, S. (1997). Long Short Term Memory. *https://www.bioinf.jku.at/publications/older/2604.pdf*.

Surve, A. R., Momin, S. M., Sawadkar, S. D., & Muskawad, A. (2020, 10 01). Amazon Review Sentiment Analysis Using FastText. *IJSRD - International Journal for Scientific Research & Development, 8*(7), 301-303. Retrieved 04 24, 2025, from https://www.ijsrd.com/articles/IJSRDV8I70205.pdf

Sutton, R. (2019, March 13). *incompleteideas.net*. Retrieved from http://www.incompleteideas.net/IncIdeas/BitterLesson.html

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., & Le, Q. V. (2020). Unsupervised Data Augmentation for Consistency Training. *Advances in neural information processing systems, 33*, 6256-6268.

Yoon, K. (2014). *Convolutional Neural Networks for Sentence Classification* (Vol. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)). Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/D14-1181

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning Based. *IEEE Computational Intelligence Magazine, 13*, 55-75. doi:10.1109/MCI.2018.2840738